Finding the Length and Midpoint of a Line Segment

AI24BTECH11021 - Manvik Muthyapu

Problem Statement

Find the length of the segment joining $\vec{A}(-6, 7)$ and $\vec{B}(-1, -5)$. Also, find the midpoint of $\vec{AB}$.

Given Points

Variable	**Description**
$\vec{A}$	-6
7	
$\vec{B}$	-1
-5	
$\vec{M}$	$\frac{\vec{A}+\vec{B}}{2}$

Length of Line Segment

The length of the line segment $\vec{AB}$ is given by $||B - A||$.

$$\begin{aligned}
B - A &= \begin{pmatrix} -1 \\ -5 \end{pmatrix} - \begin{pmatrix} -6 \\ 7 \end{pmatrix} \\
&= \begin{pmatrix} 5 \\ -12 \end{pmatrix} \\
||B - A|| &= \sqrt{(B - A)^T (B - A)} \\
&= \sqrt{5^2 + (-12)^2} = \sqrt{169} \\
&= 13
\end{aligned}$$

Thus, the length of the line segment is 13 units.

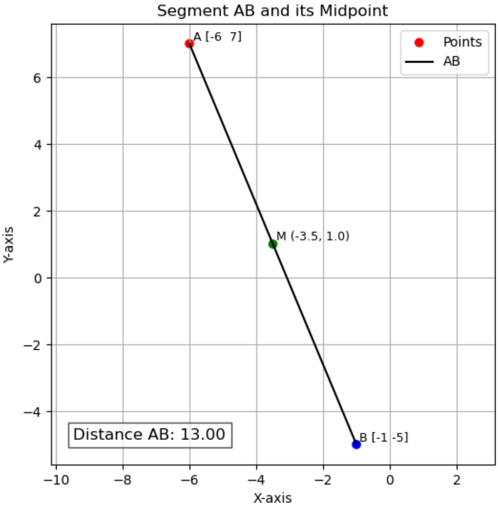
Midpoint of Line Segment

The midpoint M of line segment $\vec{AB}$ is calculated as:

$$\begin{aligned} M &= \frac{A + B}{2} \\ &= \frac{\begin{pmatrix} -6 \\ 7 \end{pmatrix} + \begin{pmatrix} -1 \\ -5 \end{pmatrix}}{2} \\ &= \frac{\begin{pmatrix} -7 \\ 2 \end{pmatrix}}{2} \\ &= \begin{pmatrix} -3.5 \\ 1 \end{pmatrix} \end{aligned}$$

Therefore, $M = \begin{pmatrix} -3.5 \\ 1 \end{pmatrix}$.

Graphical Representation



C Code

```c
#include <math.h>

// Function to calculate the distance between two points
double distance(double x1, double y1, double x2, double
y2) {
    return sqrt((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 -
    y1));
}

// Function to calculate the midpoint between two points
void midpoint(double x1, double y1, double x2, double y2,
double* mx, double* my) {
    *mx = (x1 + x2) / 2.0;
    *my = (y1 + y2) / 2.0;
}
```

Python Code

```python
import sys
sys.path.insert(0, '/home/manvik/matgeo/codes/CoordGeo')
# path to my scripts
import numpy as np
import matplotlib.pyplot as plt
import ctypes

# Load the shared library
geometry = ctypes.CDLL('./geometry.so')

# Define argument and return types for C functions
geometry.distance.restype = ctypes.c_double
geometry.distance.argtypes = [ctypes.c_double,
ctypes.c_double, ctypes.c_double, ctypes.c_double]
geometry.midpoint.argtypes = [ctypes.c_double,
ctypes.c_double, ctypes.c_double, ctypes.c_double,
ctypes.POINTER(ctypes.c_double),
ctypes.POINTER(ctypes.c_double)]
```

Python Code

```
# Calculate distance and midpoint using C functions
def calculate(A, B):
    dist = geometry.distance(A[0], A[1], B[0], B[1])
    mx, my = ctypes.c_double(), ctypes.c_double()
    geometry.midpoint(A[0], A[1], B[0], B[1],
    ctypes.byref(mx), ctypes.byref(my))
    return dist, (mx.value, my.value)

# Generate the plot
def plot(A, B, distance, midpoint):
    plt.figure(figsize=(6, 6))

    # Prepare coordinates for plotting
    x_coords = [A[0], B[0], midpoint[0]]  # x-coordinates
     ↪  for A, B, and midpoint
    y_coords = [A[1], B[1], midpoint[1]]  # y-coordinates
     ↪  for A, B, and midpoint
```

Python Code

```
# Plot points A, B, and midpoint M
plt.scatter(x_coords, y_coords, color=['red', 'blue',
'green'], label="Points")

# Connect points A and B with a line
plt.plot([A[0], B[0]], [A[1], B[1]], label="AB",
color='black')

# Labels for the points
labels = [f'A {A}', f'B {B}', f'M {midpoint}']

# Add labels to points A, B, and M
for i, (x, y) in enumerate(zip(x_coords, y_coords)):
    plt.text(x + 0.1, y + 0.1, labels[i], fontsize=9)

# Display the calculated distance on the plot (now in
 ↪  the bottom-left corner)
plt.text(0.05, 0.05, f'Distance AB: {distance:.2f}',
transform=plt.gca().transAxes, fontsize=12,
```

Python Code

```python
         bbox=dict(facecolor='white', alpha=0.7),
         verticalalignment='bottom',
         horizontalalignment='left')

# Set axis labels and title
plt.title('Segment AB and its Midpoint')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.legend()
plt.grid(True)
plt.axis('equal')
plt.show()
```

Python Code

```python
def main():
    A, B = np.array([-6, 7]), np.array([-1, -5])
    distance, midpoint = calculate(A, B)

    with open('results.txt', 'w') as file:
        file.write(f"{A[0]}, {A[1]}, {B[0]}, {B[1]},
        {distance}, {midpoint[0]}, {midpoint[1]}\n")

    plot(A, B, distance, midpoint)

if __name__ == "__main__":
    main()
```